

## Problem 1

Let

$$V(G) = \{v_1, \dots, v_n\} \quad \text{and} \quad S_n = \{\sigma : \sigma \in \{1, \dots, n\}^{\{1, \dots, n\}} \wedge \sigma \text{ is a bijection}\}.$$

Define

$$\text{cutwidth}_\sigma(i) = |\{(u, v) : u \in \{v_{\sigma_1} \dots v_{\sigma_i}\} \wedge v \in \{v_{\sigma_{i+1}}, \dots, v_{\sigma_n}\} \wedge (u, v) \in E(G)\}|,$$

where  $\sigma \in S_n$  is any permutation. The objective is to find a permutation  $\sigma \in S_n$  that minimizes the value

$$\max_{i \in \{1, \dots, n-1\}} \text{cutwidth}_\sigma(i)$$

and to calculate this minimum.

Define the function

$$\text{out}(X) = |\{(u, v) : u \in X \wedge v \in V(G) \setminus X \wedge (u, v) \in E(G)\}|,$$

where  $X$  is any subset of  $V(G)$ . Then

$$\text{cutwidth}_\sigma(i) = \text{out}(\{v_{\sigma_1}, \dots, v_{\sigma_i}\}).$$

For any given  $X$ ,  $\text{out}(X)$  can be calculated in time  $n^{\mathcal{O}(1)}$ .

We will use dynamic programming over subsets. Define

$$\text{dp}(X) = \min \left\{ \max_{i \in \{1, \dots, |X|-1\}} \text{cutwidth}_\sigma(i) : \sigma \in S_n \wedge \{v_{\sigma_1}, \dots, v_{\sigma_{|X|}}\} = X \right\},$$

where  $X$  is any subset of  $V(G)$ . Then

$$\begin{aligned} \text{dp}(\emptyset) &= 0, \\ \text{dp}(X) &= \min\{\max(\text{dp}(X \setminus \{x\}), \text{out}(X \setminus \{x\})) : x \in X\}. \end{aligned}$$

The answer to the problem is  $\text{dp}(\{1, \dots, n\})$ . To compute  $\text{dp}$  for all subsets of  $V(G)$ , we iterate over subsets in non-decreasing order of size. The time complexity of this algorithm is  $2^n \cdot n^{\mathcal{O}(1)}$ .

## Problem 2

**Lemma 1** A set of points  $S$ , in which no three points are collinear, does not form the vertices of a convex polygon if and only if there exist points  $A, B, C, D \in S$  such that  $D$  lies inside  $\triangle ABC$ .

**Proof of lemma 1** The implication „to the left” is obvious, so we only need to prove the implication „to the right”. Let  $\{H_1, \dots, H_h\}$  be the convex hull of the set  $S$ , with the assumption that these points are ordered counterclockwise along the hull. Let  $D$  be any point in  $S$  that does not belong to the hull, and let  $A = H_1$ . There exists exactly one  $i \in \{2, \dots, h-1\}$  such that points  $H_2, \dots, H_i$  lie on one side of the line  $AD$ , while points  $H_{i+1}, \dots, H_h$  lie on the other side. Taking  $B = H_i$  and  $C = H_{i+1}$ , we will obtain the desired points.

According to lemma 1, if there exist points  $A, B, C, D \in S$  such that  $D$  lies inside  $\triangle ABC$ , then at least one of these points must be removed from  $S$ . This observation leads to the following algorithm:

---

**Algorithm 1** ConvexDeletion

---

```

1: procedure CONVEXDELETION( $S, k$ )
2:   if no four points  $A, B, C, D \in S$  satisfy that  $D$  lies inside  $\triangle ABC$  then
3:     return true
4:   end if
5:   if  $k \leq 0$  then
6:     return false
7:   end if
8:   Choose points  $A, B, C, D \in S$  such that  $D$  lies inside  $\triangle ABC$ 
9:   return CONVEXDELETION( $S \setminus \{A\}, k - 1$ ) or CONVEXDELETION( $S \setminus \{B\}, k - 1$ ) or
      CONVEXDELETION( $S \setminus \{C\}, k - 1$ ) or CONVEXDELETION( $S \setminus \{D\}, k - 1$ )
10: end procedure

```

---

Finding such quadruples of points  $A, B, C, D$  can easily be done in  $\mathcal{O}(n^4)$  time by examining all quadruples of points, calculating the relevant cross products for each, and comparing their signs. This can also be done in  $\mathcal{O}(n \log n)$  time by computing the convex hull using Graham's algorithm and applying the constructive proof of lemma 1.

The depth of the recursion tree from the algorithm 1 is at most  $k$ , since with each recursive call, the parameter  $k$  decreases by 1. Each node of this tree has at most four children, which gives us an upper bound on the number of nodes in the tree:

$$\sum_{i=0}^k 4^i = \frac{4^{k+1} - 1}{3} = \mathcal{O}(4^k).$$

Therefore, the overall complexity of the algorithm 1 is  $\mathcal{O}(4^k) \cdot n^{\mathcal{O}(1)}$ .

### Problem 3

Let  $d = 10$ . We will begin by reducing the size of  $\mathcal{F}$  to a polynomial in  $k$ .

If  $|\mathcal{F}| \leq k^{d+1}$ , no reduction is necessary. Otherwise, there exists an element  $a_1 \in \bigcup \mathcal{F}$  such that the subset  $\mathcal{A}_1 = \{A \in \mathcal{F} : a_1 \in A\}$  has size at least  $k^d + 1$ . If this were not the case, a hitting set of size  $k$  would cover at most  $k^{d+1}$  sets. We either include  $a_1$  in the hitting set or exclude it. If we exclude it, then there exists an element  $a_2 \in \bigcup \mathcal{F} \setminus \{a_1\}$  such that the subset  $\mathcal{A}_2 = \{A \in \mathcal{A}_1 : a_2 \in A\}$  has size at least  $k^{d-1} + 1$ , and so on.

By repeating this process until it is possible, we obtain a set of  $l$  distinct elements  $\{a_1, \dots, a_l\}$  and a corresponding set of families  $\{\mathcal{A}_1, \dots, \mathcal{A}_l\}$  such that for each  $i \in \{1, \dots, l\}$ ,  $\mathcal{A}_{i-1} \supseteq \mathcal{A}_i$  (with  $\mathcal{A}_0 = \mathcal{F}$ ),  $|\mathcal{A}_i| \geq k^{d+1-i} + 1$ , and

$$\forall_{j \in \{i, i+1, \dots, l\}} \quad \forall_{A \in \mathcal{A}_j} \quad a_i \in A.$$

Suppose, for the sake of contradiction, that  $l > d$ . Then  $|\mathcal{A}_{d+1}| \geq 2$ , and all sets in  $\mathcal{A}_{d+1}$  would contain  $\{a_1, \dots, a_{d+1}\}$ , which contradicts the assumption that for any two distinct sets  $A, B \in \mathcal{F}$ ,  $|A \cap B| \leq d$ .

At least one element from  $\{a_1, \dots, a_l\}$  must be included in the hitting set. This can be verified, as otherwise, to cover the entire family  $\mathcal{A}_l$  (which has size at least  $k^{d+1-l} + 1$ ), there would need

to be an element  $a_{l+1}$  contained in at least  $k^{d-l} + 1$  sets from  $\mathcal{A}_l$ . This would contradict the fact that the process was repeated until it was possible.

If we include any  $a \in \{a_1, \dots, a_l\}$  in the hitting set, every set  $A \in \mathcal{A}_1$  will be covered. Therefore, we can apply the following reduction, until it is possible:

R1: If  $|\mathcal{F}| > k^{d+1}$ , find  $\{a_1, \dots, a_l\}$  and  $\{\mathcal{A}_1, \dots, \mathcal{A}_l\}$ , then replace  $\mathcal{F}$  with  $(\mathcal{F} \setminus \mathcal{A}_1) \cup \{\{a_1, \dots, a_l\}\}$ .

Note that after applying R1, the condition that for any two distinct sets  $A, B \in \mathcal{F}$ ,  $|A \cap B| \leq d$  still holds, as the newly added set is a subset of one or more sets that were originally in  $\mathcal{F}$ .

The sets  $\{a_1, \dots, a_l\}$  and  $\{\mathcal{A}_1, \dots, \mathcal{A}_l\}$  can be found in polynomial time with respect to the input size. To identify  $a_i$ , we iterate through each  $a \in \bigcup \mathcal{A}_{i-1}$ , checking which sets in  $\mathcal{A}_{i-1}$  contain it.

Applying R1 will require polynomial time overall, as each application of R1 decreases the size of  $\mathcal{F}$  by at least  $k^d + 1 - 1 = k^d > 0$ , so R1 will be applied at most  $|\mathcal{F}|$  times.

Once the size of  $|\mathcal{F}|$  has been reduced to  $\mathcal{O}(k^{d+1})$ , we still need to reduce the size of  $\bigcup \mathcal{F}$ . We apply the following reduction as long as it is possible:

R2: If there exists a set  $A \in \mathcal{F}$  such that for every  $B \in \mathcal{F} \setminus \{A\}$ ,  $A \cap B = \emptyset$ , we replace  $\mathcal{F}$  with  $\mathcal{F} \setminus \{A\}$ ,  $k$  with  $k - 1$ , and add any  $a \in A$  to the hitting set, unless  $A$  is empty, in which case we reject.

Now we assume that for any two distinct sets  $A, B \in \mathcal{F}$ ,  $A \cap B \neq \emptyset$ , which enables us to apply the next reduction until it is no longer possible:

R3: If there exists an element  $a \in \bigcup \mathcal{F}$  that is contained in only one set in  $\mathcal{F}$ , replace  $\mathcal{F}$  with  $\{A \setminus \{a\} : A \in \mathcal{F}\}$ .

This reduction is valid, as if  $a$  were required in the hitting set, it could be replaced with any other element from the set in  $\mathcal{F}$  that contains it, particularly an element included in at least two sets in  $\mathcal{F}$ .

After applying R2 and R3, every element in  $\bigcup \mathcal{F}$  is contained in at least two sets in  $\mathcal{F}$ . Therefore, the size of  $\bigcup \mathcal{F}$  is bounded by:

$$\left| \bigcup \mathcal{F} \right| \leq \frac{1}{2} \cdot \sum_{A, B \in \mathcal{F} \wedge A \neq B} |A \cap B| \leq \frac{k^{d+1}(k^{d+1} - 1)d}{2} = k^{\mathcal{O}(1)},$$

which completes the proof.

#### Problem 4

We will apply the iterative compression technique. Let

$$V(G) = \{v_1, \dots, v_n\} \quad \text{and} \quad G_i = G[\{v_1, \dots, v_i\}], \quad \text{for any } i \in \{1, \dots, n\},$$

where  $G[S]$  represents the induced subgraph of  $G$  on  $S$ . Assume we have already identified a subset  $X_i \subseteq V(G_i)$  of size at most  $k$  such that  $G_i \setminus X_i$  is a split graph. If no such subset exists for some  $i \in \{1, \dots, n\}$ , we can already reject. Initially, let  $X_1 = \emptyset$ . Our objective is to find a subset  $X_{i+1} \subseteq V(G_{i+1})$  of size at most  $k$  such that  $G_{i+1} \setminus X_{i+1}$  is a split graph.

Let  $X'_{i+1} = X_i \cup \{v_{i+1}\}$ . If  $|X'_{i+1}| \leq k$ , we can simply set  $X_{i+1} = X'_{i+1}$ . Otherwise,  $|X'_{i+1}| = k+1$ . We then iterate over each subset  $Y \subset X'_{i+1}$ , treating  $Y$  as  $X_{i+1} \cap X'_{i+1}$ . Define  $Z = X'_{i+1} \setminus Y$ ,

$W = V(G_{i+1}) \setminus X'_{i+1}$ , and  $l = k - |Y|$ . The task is to determine if we can remove at most  $l$  vertices from  $W$  to obtain  $W'$  such that  $G_{i+1}[Z \cup W']$  is a split graph.

Since the class of graphs that can be partitioned into a clique and an independent set is closed under taking subgraphs,  $G_{i+1}[Z]$  must be a split graph.  $G_{i+1}[W] = G_i[W]$  is also a split graph, because  $X_i$  is a solution for  $G_i$ . Note that we can verify if a graph is a split graph in linear time, as well as find its  $(C, I)$ -partition, where  $C$  is a clique and  $I$  is an independent set.

Let  $(C_Z, I_Z)$  be the  $(C, I)$ -partition of  $G_{i+1}[Z]$ . If no such partition exists, we move to the next subset of  $X'_{i+1}$ . The partition is not necessarily unique; this ambiguity arises when there are no edges within  $Z$  in  $G_{i+1}$ . In this case, we consider every possible choice of the clique in  $G_{i+1}[Z]$  (a single vertex or an empty set), which adds a factor of  $\mathcal{O}(k)$  to the time complexity. However, this factor is at most polynomial in  $n$ , so it does not affect the overall complexity. Henceforth, we assume  $(C_Z, I_Z)$  is unique.

Let  $(C_W, I_W)$  denote the  $(C, I)$ -partition of  $G_{i+1}[W]$ . Uniqueness here is irrelevant. Let  $(C^*, I^*)$  be the  $(C, I)$ -partition of  $G_{i+1}[Z \cup W']$  that we seek. Since  $(C_Z, I_Z)$  is unique, it must hold that  $C_Z \subseteq C^*$  and  $I_Z \subseteq I^*$ . Thus, for each  $v \in W$ , we have three choices:

1. add  $v$  to  $C^*$ ,
2. add  $v$  to  $I^*$ ,
3. remove  $v$ .

Suppose, for the sake of contradiction, that two distinct vertices  $c_1, c_2 \in C_W$  were added to  $I^*$ . Then  $G_{i+1}[I^*]$  would not be an independent set, as  $c_1$  and  $c_2$  would be connected by an edge, which is a contradiction. Therefore, at most one vertex in  $C_W$  can be added to  $I^*$ , while every other vertex in  $C_W$  must either be added to  $C^*$  or removed.

Similarly, at most one vertex in  $I_W$  can be added to  $C^*$ , while every other vertex in  $I_W$  must either be added to  $I^*$  or removed.

We can iterate through every option of adding a vertex  $v \in C_W$  to  $I^*$  (one of these options is to add no such vertex). A particular  $v \in C_W$  can be considered for addition to  $I^*$  only if  $I_Z \cup \{v\}$  forms an independent set in  $G_{i+1}$ .

Similarly, we can iterate through every option of adding a vertex  $v \in I_W$  to  $C^*$  (again, one of these options is to add no such vertex). A specific  $v \in I_W$  can be considered for addition to  $C^*$  only if  $G_{i+1}[C_Z \cup \{v\}]$  is a clique.

There are  $\mathcal{O}(|W|^2)$  such options in total, each requiring a polynomial number of edge checks. Since  $|W| \leq n$ , this factor can be ignored in the complexity analysis. Now, we assume that every  $v \in C_W$  will either be removed or added to  $C^*$ , and similarly, every  $v \in I_W$  will either be removed or added to  $I^*$ .

Let  $C'_W$  be any subset of  $C_W$ . Then  $G_{i+1}[C_Z \cup C'_W]$  is a clique if and only if

$$\forall_{u \in C_Z} \quad \forall_{v \in C'_W} \quad (u, v) \in E(G_{i+1}).$$

This holds if  $G_{i+1}[C_Z \cup C'_W]$  is indeed a clique. Conversely, if all such edges exist, then  $G_{i+1}[C_Z \cup C'_W]$  must be a clique, as both  $G_{i+1}[C_Z]$  and  $G_{i+1}[C'_W]$  are cliques. Thus, for each vertex  $v \in C_W$ , we can independently decide whether to add it to  $C^*$ , based on its connectivity to all vertices in  $C_Z$ .

Similarly, for each vertex  $v \in I_W$ , we add it to  $I^*$  only if it has no neighbors in  $I_Z$ . If, in the end, we have removed at most  $l$  vertices from  $W$ , we have found a valid solution  $X_{i+1} = Y \cup (W \setminus W')$  for  $G_{i+1}$ .

The total time complexity is  $2^k \cdot n^{\mathcal{O}(1)}$ .